МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине

‘Распределённые системы хранения данных’

Вариант №35455

Выполнил:
Студент группы P33312
Соболев Иван
Александрович
Преподаватель:
Осипов Святослав
Владимирович

УНИВЕРСИТЕТ ИТМО

Санкт-Петербург, 2024

Задание:

Цель работы - на выделенном узле создать и сконфигурировать новый кластер БД Postgres, саму БД, табличные пространства и новую роль, а также произвести наполнение базы в соответствии с заданием. Отчёт по работе должен содержать все команды по настройке, скрипты, а также измененные строки конфигурационных файлов.
Способ подключения к узлу из сети Интернет через helios:
ssh -J sXXXXXX@helios.cs.ifmo.ru:2222 postgresY@pgZZZ
Способ подключения к узлу из сети факультета:
ssh postgresY@pgZZZ
Номер выделенного узла pgZZZ, а также логин и пароль для подключения Вам выдаст преподаватель.

Обратите внимание, что домашняя директория пользователя /var/postgres/$LOGNAME

Этап 1. Инициализация кластера БД

- Директория кластера: $HOME/ifg51
- Кодировка: ISO_8859_5
- Локаль: русская
- Параметры инициализации задать через аргументы команды

Этап 2. Конфигурация и запуск сервера БД

- Способ подключения: сокет TCP/IP, принимать подключения к любому IP-адресу узла
- Номер порта: 9455
- Остальные способы подключений запретить.
- Способ аутентификации клиентов: по паролю в открытом виде
- Настроить следующие параметры сервера БД:
    - max_connections
    - shared_buffers
    - temp_buffers
    - work_mem
    - checkpoint_timeout
    - effective_cache_size
    - fsync
    - commit_delay

  Параметры должны быть подобраны в соответствии со сценарием OLAP:
  9 одновременных пользователей, пакетная запись/чтение данных по 64МБ.

- Директория WAL файлов: $HOME/krv1
- Формат лог-файлов: .csv
- Уровень сообщений лога: NOTICE
- Дополнительно логировать: контрольные точки и попытки подключения

Этап 3. Дополнительные табличные пространства и наполнение базы

- Пересоздать шаблон template1 в новом табличном пространстве: $HOME/zzo32
- На основе template1 создать новую базу: wetbluelove

- Создать новую роль, предоставить необходимые права, разрешить подключение к базе.
- От имени новой роли (не администратора) произвести наполнение ВСЕХ созданных баз тестовыми наборами данных. ВСЕ табличные пространства должны использоваться по назначению.
- Вывести список всех табличных пространств кластера и содержащиеся в них объекты.

Выполнение:

Подключение к узлу:

```
ssh -J s336760@helios.cs.ifmo.ru:2222 postgres6@pg157
```

```
PS C:\Users\Asus> ssh -J s336760@helios.cs.ifmo.ru:2222 postgres6@pg157
(s336760@helios.cs.ifmo.ru) Password:
(postgres6@pg157) Password for postgres6@pg157.cs.ifmo.ru:
[postgres6@pg157 ~]$
```

Инициализация кластера БД:

1. Создание директории кластера БД – mkdir -p $HOME/ifg51
2. Изменение системного пользователя - chown postgres6 $HOME/ifg51
3. Инициализация кластера - initdb --encoding=ISO_8859_5 --locale=ru_RU.ISO8859-5 --username=postgres6 -D $HOME/ifg51

```
[postgres6@pg157 ~]$ initdb --encoding=ISO_8859_5 --locale=ru_RU.ISO8859-5 --username=postgres6 -D $HOME/ifg51
Файлы, относящиеся к этой СУБД, будут принадлежать пользователю "postgres6".
От его имени также будет запускаться процесс сервера.

Кластер баз данных будет инициализирован с локалью "ru_RU.ISO8859-5".
Выбрана конфигурация текстового поиска по умолчанию "russian".

Контроль целостности страниц данных отключён.

исправление прав для существующего каталога /var/db/postgres6/ifg51... ок
создание подкаталогов... ок
выбирается реализация динамической разделяемой памяти... posix
выбирается значение max_connections по умолчанию... 100
выбирается значение shared_buffers по умолчанию... 128MB
выбирается часовой пояс по умолчанию... W-SU
создание конфигурационных файлов... ок
выполняется подготовительный скрипт... ок
выполняется заключительная инициализация... ок
сохранение данных на диске... ок

initdb: предупреждение: включение метода аутентификации "trust" для локальных подключений
Другой метод можно выбрать, отредактировав pg_hba.conf или используя ключи -A,
--auth-local или --auth-host при следующем выполнении initdb.

Готово. Теперь вы можете запустить сервер баз данных:

    pg_ctl -D /var/db/postgres6/ifg51 -l файл_журнала start
```

Конфигурация и запуск сервера БД:

Передача конфигурационных файлов через scp:

На узле нет vim'a, а работать в vi некомфортно. Поэтому передает на гелиос конфигурационные файлы:

scp postgres6@pg157:ifg51/pg_hba.conf .

scp postgres6@pg157:ifg51/postgresql.conf .

Обратно отправляем на узел:

scp pg_hba.conf postgres6@pg157:ifg51/pg_hba.conf

scp postgresql.conf postgres6@pg157:ifg51/postgresql.conf

pg_hba.conf

Разрешаем подключение по паролю через host, остальные способы подключения запрещаем:

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

#lab2
host    all             all             all                     password

# "local" is for Unix domain socket connections only
local   all             all                                     reject
# IPv4 local connections:
host    all             all             127.0.0.1/32            reject
# IPv6 local connections:
host    all             all             ::1/128                 reject
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     reject
host    replication     all             127.0.0.1/32            reject
host    replication     all             ::1/128                 reject
```

postgresql.conf

Меняем следующие параметры:

порт и порты слушания:

```
# - Connection Settings -

listen_addresses = '*'                  # what IP address(es) to listen on;
                                        # comma-separated list of addresses;
                                        # defaults to 'localhost'; use '*' for all
                                        # (change requires restart)
port = 9455                             # (change requires restart)
```

ставим максимальное количество подключённых пользователей равным девяти:

```
max_connections = 9                     # (change requires restart)
```

Установка размеров буферов shared_buffers, temp_buffers и work_mem:

```
# - Memory -

shared_buffers = 2GB                    # min 128kB
                                        # (change requires restart)
#huge_pages = try                       # on, off, or try
                                        # (change requires restart)
#huge_page_size = 0                     # zero for system default
                                        # (change requires restart)
temp_buffers = 256MB                    # min 800kB
#max_prepared_transactions = 0          # zero disables the feature
                                        # (change requires restart)
# Caution: it is not advisable to set max_prepared_transactions nonzero unless
# you actively intend to use prepared transactions.
work_mem = 128MB                                # min 64kB
```

shared_buffers: параметр, который задает количество памяти, которое PostgreSQL будет использовать для кэширования данных из таблиц и индексов в оперативной памяти.

Из документации PostgreSQL следует, что shared_buffers следует держать от 25% до 40% от всего выделенного ОЗУ. Так как в варианте не сказано сколько ОЗУ выделено на кластер, то будем использовать наиболее частое значение 8Гб, тогда на shared_buffers выделим 2Гб.

temp_buffers: параметр устанавливает максимальный размер оперативной памяти, которую сервер может использовать для хранения временных файлов сессии, создаваемых в процессе выполнения операций сортировки и объединения данных.

Размер временных буферов следует сделать достаточно большим(256МБ), так как система соответствует OLAP с пакетной передачей в среднем по 64 МБ.

work_mem: параметр отвечает за количество памяти, выделяемой для выполнения операций сортировки, хэширования, агрегирования и других операций обработки данных для каждой сессии.

Так как операции большие, и, скорее всего, используют множество сортировок и хэш-таблиц, то значение work_mem я выставил в 128МБ (объём памяти для внутренних операций сортировок и хэш-таблиц).

- effective_cache_size = 4GB. Оставил по умолчанию (должен быть не меньше чем shared_buffers).

- fsync = on. Оставил данный параметр включенным, чтобы запись на диск происходила. fsync: Параметр определяет, включена ли синхронизация записи на диск в PostgreSQL. Флаг fsync имеет смысл отключать на read-only копиях бд, в других случаях нужно включать для повышения отказоустойчивости независимо от конфигурации системы.

- commit_delay = 0(мс). Сохранение на WAL начинается сразу после выполнения операции. Изменение задержки перед сохранением WAL имеет смысл только в том случае, если есть возможность протестировать его влияние на общую производительность.

- checkpoint_timeout = 10min. Параметр checkpoint_timeout в PostgreSQL определяет интервал времени в секундах между запусками процесса контрольной точки (checkpoint). Поставил значение на 10 минут, так как предполагаю, что операции не частые и их немного. Не сильно отошел от стандартных 5 минут.

Включаем архивирование и указываем директорию, в которую будут копироваться WAL-файлы:

```
# - Archiving -

archive_mode = on               # enables archiving; off, on, or always
                                # (change requires restart)
archive_command = 'cp %p $HOME/krv1'            # command to use to archive a logfile segment
```

Формат лог-файлов: .csv

```
# - Where to Log -

log_destination = 'csvlog'
#log_destination = 'stderr'              # Valid values are combinations of
                                         # stderr, csvlog, syslog, and eventlog,
                                         # depending on platform.  csvlog
                                         # requires logging_collector to be on.

# This is used when logging to stderr:
logging_collector = on                   # Enable capturing of stderr and csvlog
```

Уровень сообщений лога: NOTICE

```
log_min_messages = notice               # values in order of decreasing detail:
```

Дополнительно логировать: контрольные точки и попытки подключения

```
                                        # of milliseconds.
log_checkpoints = on
log_connections = on
#log_disconnections = off
#log_duration = off
#log_error_verbosity = default          # terse, default, or verbose messages
```

Этап 3. Дополнительные табличные пространства и наполнение базы

Запускаем сервер

pg_ctl -D /var/db/postgres6/ifg51 -l logfile start

```
[postgres6@pg157 ~/ifg51]$ pg_ctl -D /var/db/postgres6/ifg51 -l logfile start
ожидание запуска сервера.... готово
сервер запущен
```

Для запуска БД необходимо сначала установить `trust` in `pg_hba.conf` и обновить дефолтный пароль для postgres6 (Я установил 123). Затем можем подключаться.

```
[postgres6@pg157 ~/ifg51]$ psql -h localhost -p 9455 -U postgres6 postgres
Пароль пользователя postgres6:
psql (14.2)
Введите "help", чтобы получить справку.

postgres=#
```

Пересоздать шаблон template1 в новом табличном пространстве: $HOME/zzo32

mkdir -p $HOME/zzo32

psql -h localhost -p 9455 -U postgres6 postgres

Создаем табличное пространство - CREATE TABLESPACE zzo32 LOCATION '/var/db/postgres6/zzo32';

Перезаписываем туда template 1:

UPDATE pg_database SET datistemplate = false WHERE datname = 'template1';

DROP DATABASE template1;

CREATE DATABASE template1 TEMPLATE template0 TABLESPACE zzo32;

UPDATE pg_database SET datistemplate = true WHERE datname = 'template1';

На основе template1 создать новую базу: wetbluelove

create database wetbluelove with template = template1;

```
postgres=# create database wetbluelove with template = template1;
CREATE DATABASE
```

Переподключаемся к базе wetbluelove:

psql -h localhost -p 9455 -U postgres6 wetbluelove

```
[postgres6@pg157 ~]$ psql -h localhost -p 9455 -U postgres6 wetbluelove
Пароль пользователя postgres6:
psql (14.2)
Введите "help", чтобы получить справку.

wetbluelove=#
```

Создать новую роль, предоставить необходимые права, разрешить подключение к базе.

create role ivan login password 'ivan';

Создадим таблицы в бд и заполним их:

psql -h localhost -p 9455 -U ivan wetbluelove -f create.sql

psql -h localhost -p 9455 -U ivan wetbluelove -f insert.sql

```
[postgres6@pg157 ~]$ psql -h localhost -p 9455 -U ivan wetbluelove -f create.sql
Пароль пользователя ivan:
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
[postgres6@pg157 ~]$ psql -h localhost -p 9455 -U ivan wetbluelove -f insert.sql
Пароль пользователя ivan:
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
```

Вывести список всех табличных пространств кластера и содержащиеся в них объекты.

select * from pg_tablespace;

```
wetbluelove=> select * from pg_tablespace;
  oid  |  spcname   | spcowner | spcacl | spcoptions
-------+------------+----------+--------+------------
  1663 | pg_default |       10 |        |
  1664 | pg_global  |       10 |        |
 16384 | zzo32      |       10 |        |
(3 строки)
```

SELECT c.relname, t.spcname FROM pg_class c JOIN pg_tablespace t ON c.reltablespace = t.oid;

```
wetbluelove=> SELECT c.relname, t.spcname FROM pg_class c JOIN pg_tablespace t ON c.reltablespace = t.oid;
                relname                 |  spcname
----------------------------------------+-----------
 pg_toast_1262                          | pg_global
 pg_toast_1262_index                    | pg_global
 pg_toast_2964                          | pg_global
 pg_toast_2964_index                    | pg_global
 pg_toast_1213                          | pg_global
 pg_toast_1213_index                    | pg_global
 pg_toast_1260                          | pg_global
 pg_toast_1260_index                    | pg_global
 pg_toast_2396                          | pg_global
 pg_toast_2396_index                    | pg_global
 pg_toast_6000                          | pg_global
 pg_toast_6000_index                    | pg_global
 pg_toast_3592                          | pg_global
 pg_toast_3592_index                    | pg_global
 pg_toast_6100                          | pg_global
 pg_toast_6100_index                    | pg_global
 pg_database_datname_index              | pg_global
 pg_database_oid_index                  | pg_global
 pg_db_role_setting_databaseid_rol_index | pg_global
 pg_tablespace_oid_index                | pg_global
 pg_tablespace_spcname_index            | pg_global
 pg_authid_rolname_index                | pg_global
 pg_authid_oid_index                    | pg_global
 pg_auth_members_role_member_index      | pg_global
 pg_auth_members_member_role_index      | pg_global
 pg_shdepend_depender_index             | pg_global
 pg_shdepend_reference_index            | pg_global
 pg_shdescription_o_c_index             | pg_global
 pg_replication_origin_roiident_index   | pg_global
 pg_replication_origin_roname_index     | pg_global
 pg_shseclabel_object_index             | pg_global
 pg_subscription_oid_index              | pg_global
 pg_subscription_subname_index          | pg_global
 pg_authid                              | pg_global
 pg_subscription                        | pg_global
 pg_database                            | pg_global
 pg_db_role_setting                     | pg_global
 pg_tablespace                          | pg_global
 pg_auth_members                        | pg_global
 pg_shdepend                            | pg_global
 pg_shdescription                       | pg_global
 pg_replication_origin                  | pg_global
 pg_shseclabel                          | pg_global
(43 строки)
```

Завершаем работу на узле, чтобы не потреблять ресурсы зря.

pg_ctl -D /var/db/postgres6/ifg51 stop

```
[postgres6@pg157 ~]$ pg_ctl -D /var/db/postgres6/ifg51 stop
ожидание завершения работы сервера.... готово
сервер остановлен
```

Выводы

Во время выполнения лабораторной работы я научился создавать и конфигурировать кластер БД PostgreSQL. Я познакомился с созданием и работой табличных пространств и ролей.